Diagramr

An Automatic Shear Force and Bending Moment Diagrammer

Alex Burka
aburka1@sccs.swarthmore.edu

Fall 2011

Abstract

Shear force and bending moment diagrams are essential tools in the design and analysis of loaded beams, aiding in the calculation of maximum stress and deflection, which determines whether a structure will fail under load. However, the process for drawing these diagrams by hand is tedious, involving multiple integrations and free-body diagrams. Following Jonathan Sarmiento [1], I demonstrate a computer interface for arranging beams and loadings, which then automatically draws the corresponding shear force and bending moment diagrams. Chief benefits are accuracy and robustness of implementation, ease of use of the interface, and portability across the major operating systems (Windows, Mac OS X, and Linux). The program is web browser-based, supporting Mozilla Firefox [3] and Google Chrome [4], but not older versions of Internet Explorer. The application is online at http://sccs.swarthmore.edu/~aburka1/diagramr.

Contents

1 Rationale and Theory . . . 3
1.1 Transverse Beam Loading: Shear Force and Bending Moment . . . 3
1.2 Singularity Functions . . . 3
2 Software Design and Implementation . . . 3
2.1 Languages . . . 3
2.2 Libraries Used . . . 4
2.3 Components . . . 4
2.4 Data Flow . . . 6
2.5 Portability . . . 6
3 User Guide and Examples . . . 7
3.1 Interface Elements . . . 7
3.2 Examples . . . 8
4 Conclusion . . . 10
5 References . . . 12
6 Code Listing . . . 12
6.1 index.html . . . 12
6.2 diagramr.css . . . 13
6.3 beam.js . . . 14
6.4 interface.js . . . 20
6.5 graphics.js . . . 32
6.6 oops.js . . . 46
6.7 raphael-ext.js . . . 48
7 Final Presentation Slides . . . 51

List of Figures

1 Data flow from beam to graphs . . . 7
2 Example 5.4 from Beer et al, p. 334 . . . 9
3 Example 5.5 from Beer et al, p. 335 . . . 9
4 Example 5.6 from Beer et al, p. 335 . . . 10
5 Example 5.3 from Beer et al, p. 333 . . . 10

List of Tables

1 Singularity function integration rules . . . 3

1 Rationale and Theory

1.1 Transverse Beam Loading: Shear Force and Bending Moment

Many simple problems in structural engineering can be modeled as a beam with transverse loading. For example, a bridge supported at its end can be represented as a simply supported beam. A flagpole in windy conditions is a cantilevered beam, as long as gravity can be neglected. In all situations with transverse beam loading, it is desirable to find the internal forces in the member under consideration, in order to determine the deflection and anticipate failures due to excessive stress or strain. Essential tools in this process are the shear force and bending moment diagrams, which show the internal forces and moments at every point within the member. Learning how to draw these diagrams is important in gaining an understanding of solid mechanics, and as a foundation for further study. Diagramr is an aid in making simple shear force and bending moment diagrams corresponding to a given loading situation. It is designed to help in teaching the subject, and also to help the engineer in calculation. Diagramr will not solve the most complicated problems, but it can be used to check one's understanding and confirm hypotheses.

The computational process for plotting the shear force and bending moment from a given beam loading is relatively simple. First the loading of the beam, including reactions, must be expressed as a single mathematical function of x (1). Then, the shear force is simply the integral of this function (2), and the bending moment is the integral of the shear force (3). In order to conveniently represent these (often discontinuous) functions as single expressions, and avoid constants of integration, we use singularity functions.

$$w(x) = f(x) \tag{1}$$

$$V(x) = \int w(x)dx \tag{2}$$

$$M(x) = \int V(x)dx \tag{3}$$

1.2 Singularity Functions

Singularity functions are a technique for represent discontinuous functions as a single expression [11]. Specifically, a singularity function term is a binomial multiplied by a step function, representing the fact that it starts a certain point along the x axis. They are written as $m\langle x-a\rangle^n$, which represents $m(x-a)^n u(x-a)$, where $u(x)$ is the Heaviside step function. Table 1 summarizes the integration rules, which do not require constants of integration.

$f(x)$	definition	$\int f(x)dx$
$m\langle x-a\rangle^n,\ n<0$	$m\frac{d^{n+1}}{dx^{n+1}}\delta(x-a)$	$m\langle x-a\rangle^{n+1}$
$m\langle x-a\rangle^n,\ n\geq 0$	$m(x-a)^n u(x-a)$	$\frac{m}{n+1}\langle x-a\rangle^{n+1}$

Table 1: Singularity function integration rules

2 Software Design and Implementation

2.1 Languages

Diagramr is implemented as web application, using the standard technologies of the modern World Wide Web. The layout of the interface is written in HTML and CSS, but this layout is very simple and does not include any of the graphics elements. All of the graphics and interactivity are done

in JavaScript. The mathematics was first implemented in Clojure [9], which is a general-purpose Lisp-inspired language that runs on the Java Virtual Machine. It was to form the server backend of Diagramr. However, testing showed that modern browsers are quite fast enough to perform the calculations required for Diagramr, so the Clojure implementation was rewritten in JavaScript, and the application runs entirely on the client side.

2.2 Libraries Used

Raphaël.js

Raphaël, written and distributed by Dmitry Baranovskiy, is a "small JavaScript library that should simplify your work with vector graphics on the web," according to its website [6]. Vector graphics are a powerful technique for delivering graphics via the Internet, because the relationships between graphical elements are specified not in terms of exact pixel measurements, but by percentages and ratios. This allows the graphics to scale to the available screen size. Raphaël implements the Scalable Vector Graphics (SVG) standard [7], which is portable and easy to work with. The library also provides layers of abstraction for animation, user interactivity with events, and graphing (with the g.Raphaël extension). These abstraction layers reduce work for the programmer by obviating common boilerplate code and implementing cross-browser compatibility.

Oops.js

Oops.js is a library that I wrote in the course of implementing this project. It provides undo/redo functionality for web applications written in JavaScript. Undo and redo is an important feature for interactive systems. It is usually taken for granted in programs such as word processors and graphics editors, but absent in many web applications. I evaluated a freely available library, jsUndoable (`http://jscott.me/jsundoable.html`), but determined that writing my own would give me more control over the functionality and size of the code. The hard part of implementing an undo feature is not the undo library itself, but in structuring the rest of the application so that all actions are reversible.

The Oops.js library uses the command pattern [10], which allows the code performing actions to be decoupled from the undo/redo system. The chief restriction is that any code that affects the beam state (such as adding a new load, or moving a support) does not perform any modifications directly. Instead, it registers an "action" with the undo manager, consisting of the code to be run as well as code to undo it. The undo manager stores these actions in a stack, and exposes undo/redo calls to navigate through.

2.3 Components

Interface and Graphics

The design goals for the interface were to be fluid and as intuitive as possible. "Intuitive" is a funny concept in software engineering, but it usually refers to the fact that interfaces are easy to use if they are analogous to other interfaces which the user is familiar with, and faithful to the real-world system that they represent. Also, the interface should be predictable and consistent (the principle of least surprise).

To that end, Diagramr's interface is designed to look and feel like the process of drawing a beam loading diagram on paper. Most elements are draggable and resizeable (where applicable), and all of the numbers on the diagram can be clicked and changed. The graphics and interactivity are realized through use of the Raphaël library, which provides primitives for geometric shapes and an abstraction layer for handling JavaScript events (such as click and drag).

At runtime, the beam is represented by its Raphaël object, which corresponds to the graphical element on the screen. The URL is used for local storage of the beam configuration, so that the state of the application can be preserved across page reloads. Raphaël allows for grouping a set of objects together (in this case, the beam plus loads and supports) and for adding arbitrary data to an object (I use this facility for storing the beam length, overhanging vs. cantilever mode, and more). The graphics engine extends Raphaël with the shapes of the various loads and supports (by grouping together primitives such as lines and triangles). Each load has a "mini" version (used for the top menu) and a larger version that sits on the beam and is draggable. When the state of the beam is needed for graphing, the code walks through the Raphaël object and encodes all of the information about the beam into a compact JSON representation (see Data Flow below).

Beam physics

The beam engine (implemented in `beam.js` contains enough physics logic to reason about the distribution of loads on a given beam, calculate the reactions, and correctly arrange the terms in the singularity function describing the beam. The singularity function is then handed off to the graphics code. The functions available in the beam engine are:

- Functions operating on the entire beam
  - `reactions`: This calculates the reactions on an overhanging beam, by doing a sum of vertical forces and a sum of moments about the right support.

$$+\uparrow \quad \sum F_y = 0 = R_A + R_B + \sum_{\text{loads}} mag(L) \tag{4}$$

$$+\text{CCW} \sum M_B = 0 = R_A(x_B - x_A) + \sum_{\text{loads}} mag(L) * (centroid(L) - x_B) \tag{5}$$

    Solving these equations, we have

$$R_B = \frac{1}{x_B - x_A} \sum_L mag(L) * (centroid(L) - x_B) \tag{6}$$

$$R_A = \sum_L mag(L) - R_B \tag{7}$$

    For the purposes of finding the reactions, a few temporary transformations are made to the beam, to avoid issues. Concentrated couples are transformed into "real" couples of an upward and downward concentrated load, 0.1 meters apart (centered at the position of the concentrated couple) and each with magnitude $P = 10M$. This allows them to count in the reaction calculations without further modifications to the equations. Also, distributed loads are split at the supports, so that they can be partially counted on both sides of the support in the moment sum calculation.
  - `support2load`: Once the reactions are calculated, the supports are nothing more than concentrated loads with a known magnitude (pointing upwards). For an overhanging or simply supported beam, this function removes the supports and replaces them with concentrated loads. (For a cantilever beam, the reactions are unnecessary for the singularity function, so the support is simply removed).
  - `singularity`: This function creates a singularity function for an entire beam. The beam should already have been transformed by `support2load` so that the supports can be included in the singularity function, if necessary. It uses the `singularity_function` defined below.
- Functions operating on individual loads

All of these functions are multiplexed on load type.

  - `magnitude`: Return the magnitude of a given load. For a concentrated load or couple, this is simply the magnitude, while for a distributed load it is the area under the pressure curve.
  - `centroid`: Return the centroid (point of application) of a given load. For a concentrated load or couple, this is simply the position, while for a distributed load it is the centroid of the area under the pressure curve.
  - `singularity_function`: Construct the singularity function term for a load. Singularity function terms are represented internally as a 3-tuple of (magnitude, start, power) which corresponds to $(m, a, n) = m \langle x - a \rangle^n$. The term returned from this function is for the load function $w(x)$, so integration is necessary to get the shear force $V(x)$ or the bending moment $M(x)$.
  - `counterbalancers`: Turn a "finite" distributed load into a series of "infinite" distributed loads. Traditional singularity functions cannot represent terms that end at a specific point – only terms that start and continue forever. Therefore, to incorporate distributed loads that do have an endpoint, it is necessary to use several distributed loads that all continue towards infinity, but cancel each other out after the desired end point. This function calculates any extra distributed loads that might be needed to cancel out an infinitely extended version of the passed load.

- Miscellaneous
  - `left`: Check whether one beam element is entirely to the left of another – one or both may be distributed loads.
  - `integrate`: This function takes a singularity function (an array of singularity function term tuples) and integrates it according to the rules given in Table 1.

2.4 Data Flow

When the shear force and bending moment diagrams are redrawn, which happens on page reload or whenever a change is made that affects the beam, the beam is re-encoded into JSON and fed through the physics functions to eventually end up with the singularity function expressions for shear force and bending moment. Figure 1 shows a schematic flow of information through the system. First, the beam is transformed into a JSON descriptor that contains all the information about the beam and its loads and supports. This information is used to calculate the reactions, and then the supports are removed. In overhanging mode, the supports are replaced with concentrated loads pointing upwards. In addition, any distributed loads are converted to "infinite" loads that have a starting point but no ending point. This normalized beam state is then expressed as a singularity function (represented as a list of singularity terms). Integrating this twice yields the singularity functions for shear force and bending moment. The graphing code steps through 500 points along the x axis where it evaluates both functions, and then hands the points off to the g.Raphaël graphing library.

2.5 Portability

The web application was developed using Firefox on Linux. It has been demonstrated to work on the following platforms:

- Mozilla Firefox, on Windows 7, recent versions of Linux, and Mac OS X 10.6
- Google Chrome, on the same platforms
- Opera Mobile, on Google Android 2.3 (the stock Android browser has no SVG support)

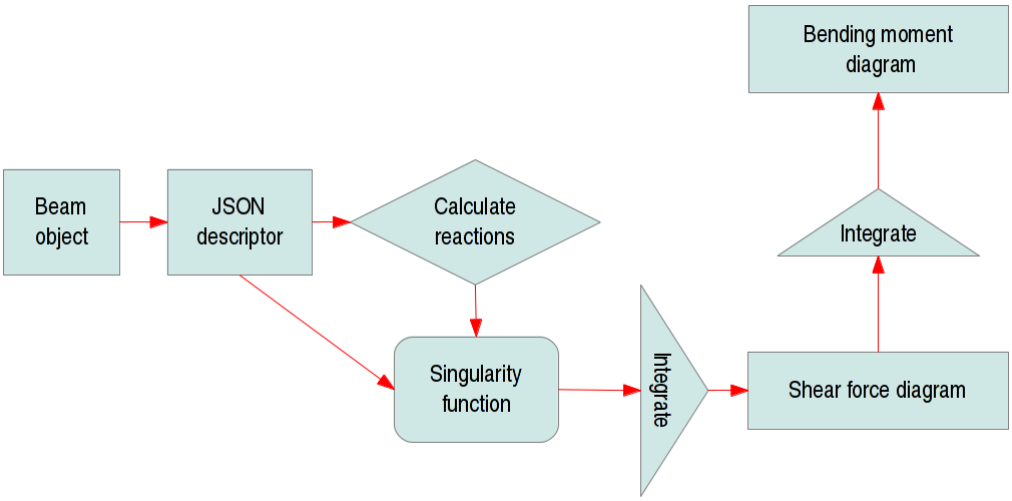


Figure 1: Data flow from beam to graphs

Functionality is greatly reduced in Internet Explorer 8, but modern standards-compliant alternatives are available, so this is not a large handicap.

3 User Guide and Examples

3.1 Interface Elements

- Menu

  The menu contains two buttons: Clear and Print. Clear simply clears all of the loads from the beam. This is the only action which is not reversible! Print is for making a hard copy of the loading diagram and the graphs. It hides the menu, the rounded border around the diagrams, and invokes the browser's print dialog.

- Adding and removing loads

  At the top of the loading diagram, just below the menu, is a selection of load types that can be added to the beam. To add a load click and drag it to a location just above the beam (or on top of it). The load will attach itself to the beam. It can then be moved and resized. Also, the shear force and bending moment diagrams will immediately be updated.

- Manipulating loads and supports on the beam

  All beam elements (including loads and supports) can be moved left and right by clicking and dragging. Point elements (supports, concentrated loads and concentrated couples) can be moved by clicking on the number right under the element and changing it in the prompt window that pops up. Distributed loads have three number underneath them. On the left is the start position (with ">"), on the right is the end position (with "<"), and in the middle is the length (with "||"). All of these can be clicked and changed (the start and end positions have syntax indicating whether the user wishes to preserve the length of the load, which is detailed in the prompt). In addition, distributed loads can be resized by clicking and dragging on their edges. Finally, all of the loads have magnitudes that are editable by clicking on the number and changing it in the prompt. For concentrated loads, the value is the force; for concentrated

couples, is it is the moment; for constant distributed loads it is the force per unit distance; and for linear distributed loads it is the slope of the load.

- Manipulating the beam
  - Beam length

    The beam length is indicated by the number on the right of the diagram, which is editable. When changing the length of the beam, the user must choose whether to add length to the left, right, or the middle of the beam, and whether to scale existing load/support positions with the length change or to keep them absolutely (the syntax for these choices is explained in the prompt).
  - Beam mode

    The beam has two available modes: overhanging and cantilever. In overhanging mode, there are two supports: one pinned and one roller. (If the supports are at the extreme ends of the beam, then the beam is technically simply supported as opposed to overhanging, but for all intents and purposes they are equivalent.) The supports are movable, and their reactions are calculated for the diagram output. In cantilever mode, the beam is supported at one end by attachment to a wall. The beam mode is indicated at the left side of the loading diagram, and can be toggled by clicking on the label (even though the two supports are hidden in cantilever mode, they are remembered if the mode is changed back to overhanging).
- Output Diagrams

  On the lower half of the screen, the shear force and bending moment diagrams are drawn. They are updated every time the state of the beam changes. In order to achieve smooth curves, 500 points distributed along the length of the beam are drawn. In addition, a zero line is drawn on both graphs for reference.

3.2 Examples

Sample problems 5.4, 5.5, 5.6 from *Mechanics of Materials*

These examples will demonstrate the correctness of the implementation and the method for entering loading diagrams into Diagramr. The examples used are from the textbook used in Swarthmore's ENGR 059 course, Mechanics of Materials [2].

5.4 (p. 334) Constant distributed load

This problem contains a 9-meter-long simply supported beam AC with a distributed load of 20 kN/m from A to B, where B is 6 meters from the left side of the beam. To represent this in Diagramr, we first change the beam length to 9 meters (L=9). This takes care of moving the right support. Then, we create a constant distributed load, and change its parameters so that it starts at 0 m, has a length of 6 m, and a magnitude of 20000 N/m. At this point the reactions are calculated and the graphs automatically drawn (see Figure 2 below). Consultation of the textbook will confirm that the answer is correct.

5.5 (p. 335) Triangular load

This problem is about a cantilever beam AC of length L with the support at the right and a triangular distributed load from A to B, where point B is located at a distance a from the left side. The distributed load starts at a height of w_0 and slopes downwards. We need concrete numbers for Diagramr, so we will arbitrarily set $L = 1$ m, $a = 0.6$ m, and $w_0 = 0.5$ N. First, click on "overhanging" to change the beam mode to cantilever. Then create a triangular load. Drag the left side to the left end of the beam, set the length to 0.6 m and the magnitude to

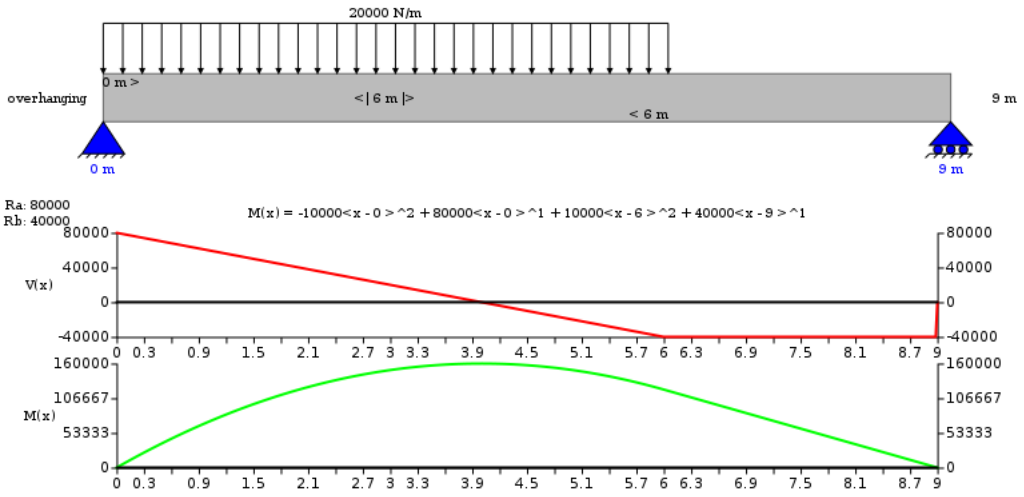


Figure 2: Example 5.4 from Beer et al, p. 334

$-\sqrt{(0.6\text{ m})^2 + (0.5\text{ N})^2} = -0.781$ N (since we are controlling the slope of the load, not the value at its end). The result is shown in Figure 3.

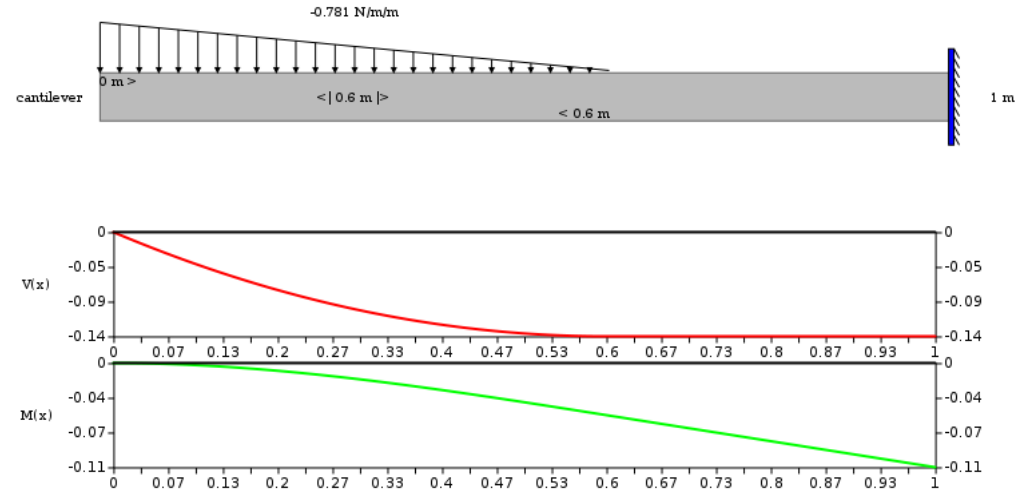


Figure 3: Example 5.5 from Beer et al, p. 335

5.6 (p. 335) Concentrated couple

In a third problem with a single, load we have a simply supported beam AC of length L with a counterclockwise concentrated couple of magnitude T at point B, which is located at a distance a from the left support. We set $L = 2$ m, $a = 0.4$ m and $T = 1$ Nm. The beam length can be changed to 2 meters (L=2) by clicking on the 1 m length. Then, drop a concentrated couple onto the beam, and move it to 0.4 m. Counterclockwise couples are positive per the right-hand rule, so no further modification is necessary. The diagrams are automatically drawn (Figure 4, and the textbook confirms their accuracy.

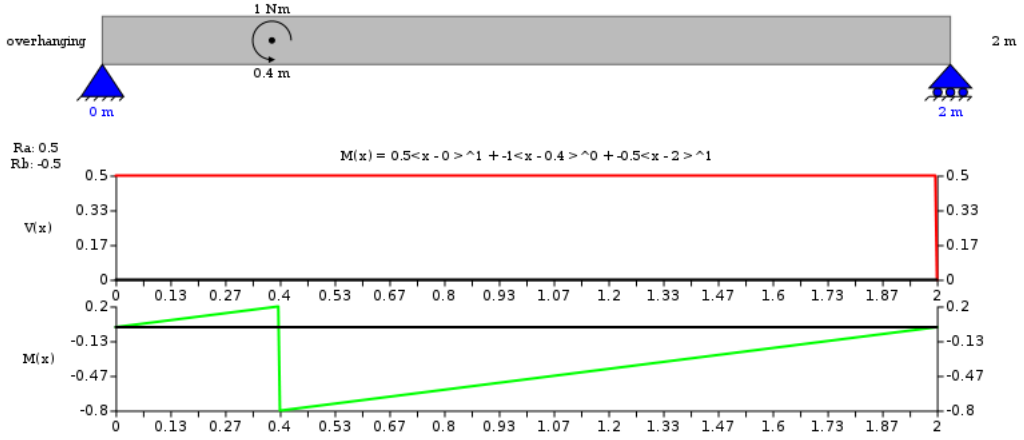


Figure 4: Example 5.6 from Beer et al, p. 335

Sample problem 5.3 (p. 333) from *Mechanics of Materials*

Here we have a more exciting problem: an overhanging beam with several loads. Unfortunately, the units are imperial, but we can simply treat feet as meters and kips as kN, and the numbers will come out right. The problem contains a 32-meter beam AE with supports at A and D (D is located 8 meters from the right end), concentrated loads at B (6 meters from the left, 20 kN) and C (14 meters from the left, 12 kN), and a distributed load of 1.5 kN/m from D to E. Diagramr can represent this in overhanging beam mode, and the answer matches the textbook, as shown below in Figure 5.

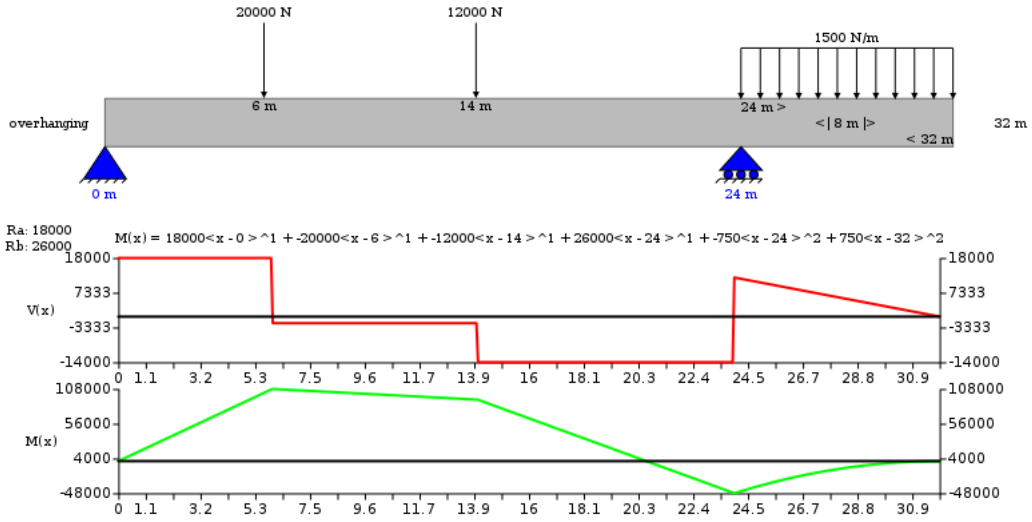


Figure 5: Example 5.3 from Beer et al, p. 333

4 Conclusion

I have demonstrated Diagramr, a web application which provides an interface for editing transverse beam loading diagrams, and automatically draws shear force and bending moment diagrams. The interface is intuitive and appropriate for students beginning to study solid mechanics. The underlying mathematics uses singularity functions to represent the integral relationships between loading, shear

force, and bending moment.

Diagramr is online at http://sccs.swarthmore.edu/~aburka1/diagramr, hosted by the Swarthmore College Computer Society (SCCS) [5]. Any questions should be directed to Alex Burka, aburka1@sccs.swarthmore.edu.

5 References

[1] Jonathan Sarmiento, *ENGR 059 Project Report: Automatic Shear Force and Bending Moment Diagrams*. Swarthmore College, 2003. 1

[2] Beer, Ferdinand P., E. R. Johnston, J. T. Dewolf, and David F. Mazurek. Mechanics of Materials. 6th ed. New York, NY: McGraw Hill, 2012. 8

[3] Mozilla Firefox, http://www.mozilla.org/en-US/firefox/fx/ 1

[4] Google Chrome, http://www.google.com/chrome 1

[5] Swarthmore College Computing Society (SCCS), http://sccs.swarthmore.edu/ 11

[6] Raphaël – JavaScript Library, http://raphaeljs.com 4

[7] Scalable Vector Graphics (SVG) 1.1 (Second Edition), http://www.w3.org/TR/SVG/ 4

[8] JSON, http://www.json.org/

[9] Clojure - home, http://clojure.org/ 4

[10] Command pattern, http://en.wikipedia.org/wiki/Command_pattern 4

[11] Singularity function, http://en.wikipedia.org/wiki/Singularity_function 3

6 Code Listing

6.1 index.html

```
<html>
  <head>
    <title>Diagramr</title>

    <link rel="stylesheet" type="text/css" href="css/diagramr.css"/>
    <script type="text/javascript" src="js/raphael.js"></script>
    <script type="text/javascript" src="js/g.raphael.js"></script>
    <script type="text/javascript" src="js/g.line.js"></script>
    <script type="text/javascript" src="js/raphael-ext.js"></script>
    <script type="text/javascript" src="js/oops.js"></script>
    <script type="text/javascript" src="js/beam.js"></script>
    <script type="text/javascript" src="js/graphics.js"></script>
    <script type="text/javascript" src="js/interface.js"></script>
  </head>
  <body>
    <div id="wrapper">
      <div id="header">
        <h1>Diagramr</h1>
        <h2>A Web-based Automatic Shear Force and Bending Moment Diagrammer"</h2>
      </div>
      <div id="interface">
        <div id="menu">
          <input type="button"
                 value="Clear"
                 style="float: left;
```

```
                        margin-top: .3em;
                        margin-left: 1em;
                        background: white"
                onclick="location.search = '';"
                value="Clear"/>
          <input type="button"
                value="Print"
                onclick="print(); return false;"
                style="float: right;
                        margin-top: .3em;
                        margin-right: 1em;
                        background: white"/>
      </div>
      <div id="io">
        <div id="input"></div>
        <div id="output"></div>
      </div>
    </div>
  </body>
</html>
```

6.2 diagramr.css

```
#header {
    position: absolute;
    top: 0px;
    left: 1em
}

#interface {
    position: absolute;
    top: 8em;
    left: 1em;

    height: 70%;
    width: 75%
}
@media screen
{
  #interface div,canvas {
      border: 1px solid black;
      border-radius: 15px
  }
}
@media print
{
  #interface #menu {
    display: none
  }
}
#interface div div {
    border: none
}
#interface #menu {
    margin-bottom: 1em;
    height: 2em;
    width: 100%;
    background-color: green
}
#interface #io {
    float: left;
```

```
39
40      height: 100%;
41      width: 100%
42  }
43  #interface #io #input,#output {
44      width: 100%;
45      height: 49%
46  }
```

6.3 beam.js

```
1   /* Diagramr beam encoding/processing functions
2    *  (this is where the mechanical engineering happens)
3    *
4    * Alex Burka, October 2011
5    */
6
7   // like map, but the function returns an array instead of
8   // a single transformed element. all of the returned arrays
9   // are concatenated.
10  Array.prototype.mapcat = function (fn)
11  {
12    return [].concat.apply([], this.map(fn));
13  }
14
15  Beam = {
16
17    // verify that a JSON object is a correctly encoded beam
18    //  return true or undefined
19    // (currently unused)
20    verify: function (beam)
21    {
22      return beam.length
23        && beam.supports
24        && beam.loads
25
26        && beam.supports.length == 2
27        && beam.supports.filter(function (s) { return s.type == "pinned"; }).length == 1
28        && beam.supports.filter(function (s) { return s.type == "roller"; }).length == 1;
29    },
30
31    // get the magnitude of any load type
32    magnitude: function (load)
33    {
34      switch (load.type)
35      {
36        case "concentrated":
37        case "moment":
38          return load.magnitude;
39        case "distributed-constant":
40          return load.magnitude * (load.end - load.start);
41        case "distributed-linear":
42          return 0.5 * Math.pow(load.end - load.start, 2) * load.magnitude;
43        default:
44          throw "unsupported load type!";
45      }
46    },
47
48    // get the centroid of any load type
49    centroid: function (load)
50    {
```

```
    switch (load.type)
    {
      case "concentrated":
      case "moment":
        return load.position;
      case "distributed-constant":
        return (load.start + load.end) / 2;
      case "distributed-linear":
        return load.start + (load.end - load.start)*2/3;
      default:
        throw "unsupported load type!";
    }
  },

  // turn a finite distributed load into a series of infinite distributed loads
  counterbalancers: function (load)
  {
    switch (load.type.substr(12))
    {
      case "constant":
        return [{type: "distributed-constant",
                 position: load.start,
                 magnitude: load.magnitude},
                {type: "distributed-constant",
                 position: load.end,
                 magnitude: -load.magnitude}];
      case "linear":
        if (load.magnitude > 0)
        {
          return [{type: "distributed-linear",
                   position: load.start,
                   magnitude: load.magnitude},
                  {type: "distributed-linear",
                   position: load.end,
                   magnitude: -load.magnitude},
                  {type: "distributed-constant",
                   position: load.end,
                   magnitude: load.magnitude * (load.start - load.end)}];
        }
        else
        {
          return [{type: "distributed-constant",
                   position: load.start,
                   magnitude: load.magnitude * (load.start - load.end)},
                  {type: "distributed-linear",
                   position: load.start,
                   magnitude: load.magnitude},
                  {type: "distributed-linear",
                   position: load.end,
                   magnitude: -load.magnitude}];
        }
      default:
        throw "unsupported load type!";
    }
  },

  // make a term of the singularity function for the load
  // m*<x - a>^n => [m, a, n]
  singularity_function: function (load)
  {
    switch (load.type)
```

```
    {
      case "concentrated":
        return [load.magnitude, load.position, -1];
      case "moment":
        return [load.magnitude, load.position, -2];
      case "distributed-constant":
        return [load.magnitude, load.position, 0];
      case "distributed-linear":
        return [load.magnitude, load.position, 1];
      default:
        throw "unsupported load type!";
    }
  },

  // is thing A entirely to the left of thing B?
  left: function (A, B)
  {
    return or(A.position, A.end) <= or(B.position, B.start);
  },

  // calculate the reactions for an overhanging beam
  reactions: function (beam)
  {
    if (beam.mode == "cantilever") return;

    function sum (arr) { return arr.reduce(function (a,b) { return a+b; }, 0); }

    // make sure the supports are in order
    beam.supports.sort(function (a, b)
                       {
                         if (a.position == b.position)
                           return 0;

                         if (a.position < b.position)
                           return -1;

                         return 1;
                       });

    // convert moments to couples, split distributed loads at supports
    var tr_loads = beam.loads.mapcat(
                     function (load)
                     {
                       // moment -> couple
                       if (load.type == "moment")
                       {
                         return [{type: "concentrated",
                                  position: load.position - 0.05,
                                  magnitude: load.magnitude*10},
                                 {type: "concentrated",
                                  position: load.position + 0.05,
                                  magnitude: -load.magnitude*10}];
                       }
                       else if (load.type.indexOf("distributed-") == 0)
                       {
                         // let's see if we have to split this load around any supports

                         var loads = [];
                         var load_start = load.start, lastmag = 0;
                         // load_start is the continually-updated position
                         //   of the start of the load that is being split
```

```
                    // lastmag is the instantaneous magnitude of the load
                    //   at the last place where it was split
                    //   (used for adding constant loads to fix up
                    //    split linear loads)

                    if ( load.type.indexOf("linear") != -1
                      && load.magnitude < 0)
                    {
                      // negatively sloped loads need to start with a lastmag
                      lastmag = load.magnitude * (load.start - load.end);
                    }

                    for (var s in beam.supports)
                    {
                      if (beam.supports.hasOwnProperty(s))
                      {
                        if (load_start < beam.supports[s].position
                         && load.end > beam.supports[s].position)
                        {
                          // load from the last split position to here
                          loads.push({type:      load.type,
                                      magnitude: load.magnitude,
                                      start:     load_start,
                                      end:       beam.supports[s].position});
                          if (lastmag > 0)
                          {
                            // we might need a constant load
                            loads.push({type:      "distributed-constant",
                                        magnitude: lastmag,
                                        start:     load_start,
                                        end:       beam.supports[s].position});
                          }

                          if (load.type.indexOf("linear") != -1)
                          {
                            // update last mag for linear loads
                            lastmag += load.magnitude
                                     * (beam.supports[s].position - load_start);
                          }
                          // update starting position, since the load was just split
                          load_start = beam.supports[s].position;
                        }
                      }
                    }

                    // in case the load extends past the last support,
                    // there might be one dangling portion to add on
                    if (load_start < load.end)
                    {
                      loads.push({type: load.type,
                                  magnitude: load.magnitude,
                                  start: load_start,
                                  end: load.end});
                      if (lastmag > 0)
                      {
                        loads.push({type:      "distributed-constant",
                                    magnitude: lastmag,
                                    start:     load_start,
                                    end:       load.end});
                      }
                    }
```

```
                        console.log(JSON.stringify(load));
                        console.log(JSON.stringify(loads));
                        return loads;
                      }
                      else
                      {
                        return [load];
                      }
                    });

    // calculate reactions
    var A = beam.supports[0],
        B = beam.supports[1],

      loads = sum(
            tr_loads
              .map(function (l) { return Beam.magnitude(l); })),
      loads_left = sum(
            tr_loads
              .filter(function (l) { return Beam.left(l, A); })
              .map(function (l) { return (Beam.magnitude(l)
                                          * (A.position - Beam.centroid(l))); })),
      loads_right = sum(
            tr_loads
              .filter(function (l) { return Beam.left(A, l); })
              .map(function (l) { return (Beam.magnitude(l)
                                          * (Beam.centroid(l) - A.position)); })),

      Rb = (loads_right - loads_left) / (B.position - A.position),
      Ra = (loads - Rb);

    return [Ra, Rb];
  },

  // given a beam with loads and supports, turn the supports into negative loads
  support2load: function (beam, reactions)
  {
    if (beam.mode == "overhanging")
    {
      // turn the supports into loads
      for (var i in reactions)
      {
        if (reactions.hasOwnProperty(i))
        {
          beam.loads.push({type: "concentrated",
                           position: beam.supports[i].position,
                           magnitude: -reactions[i]});
        }
      }
    }

    delete beam.supports;

    return beam;
  },

  // given a list of loads, construct the singularity function
  singularity: function (loads)
  {
    var inf_loads = loads.mapcat(function (l)
```

```
                                    {
                                      if (l.type.indexOf("distributed-") == 0)
                                      {
                                        return Beam.counterbalancers(l);
                                      }
                                      else
                                      {
                                        return [l];
                                      }
                                    }),
        sorted_loads = inf_loads.sort(function (a, b)
                                       {
                                         return a.position > b.position;
                                       });

      return sorted_loads.map(Beam.singularity_function);
    },

    // integrate a series of singularity-function terms
    integrate: function (terms)
    {
      return terms.map(function (sing)
                       {
                         var m = sing[0], a = sing[1], n = sing[2];

                         if (n < 0)
                         {
                           return [m, a, n+1];
                         }
                         else
                         {
                           return [m / (n+1), a, n+1];
                         }
                       });
    }
  };

  // encode the current beam into JSON format
  //  this is a recursive function
  //    so it may be called on the beam, or on a support/load
  Raphael.st.encode = Raphael.el.encode = function () {
    var parts = this.type.split("_");
    var data = {};
    switch (parts[0]) // what am I?
    {
      case "set": // I am a beam. setup the whole JSON structure
        data.length = beam.len(); // beam length in meters
        data.mode = beam.data("mode");
        data.supports = [];
        data.loads = [];
        for (var i = 0; i < this.length; ++i) // loop through everything and recurse
        {
          if (this[i].type.indexOf("support") == 0)
          {
            data.supports.push(this[i].encode());
          }
          else if (this[i].type.indexOf("load") == 0)
          {
            data.loads.push(this[i].encode());
          }
        }
```

```
      break;
    case "support": // I am a support. return a JSON fragment
      data.type = parts[1];
      data.position = this.data("position");
      break;
    case "load": // I am a load. return a JSON fragment
      data.type = parts[1];
      data.magnitude = this.data("mag");
      switch (parts[1])
      {
        case "concentrated":
        case "moment":
          data.position = this.data("position");
          break;
        case "distributed-constant":
        case "distributed-linear":
          data.start = this.data("start");
          data.end = this.data("end");
          break;
        default:
          console.log("unsupported load type!");
      }
      break;
    default:
      console.log("what am I?");
  }
  return data;
};
```

6.4 interface.js

```
/* Diagramr GUI
 *
 * Alex Burka, October 2011
 */

function draw()
{
  function doit()
  {
    var json = beam.encode(),

        reactions = Beam.reactions(json),
        fixed = Beam.support2load(json, reactions),
        loading = Beam.singularity(fixed.loads),
        shear = Beam.integrate(loading),
        moment = Beam.integrate(shear);

    draw_singularity([shear, moment], json.length, reactions, ["V(x)", "M(x)"]);
  }

  output.rect(0, 0, output.width, output.height)
    .attr({"fill": "white",
           "stroke": "white",
           "opacity": 0.5});

  setTimeout(doit, 1);
}

// these are the coordinates of the beam, expressed as percentages of the paper size
var BEAM_X = 0.1;
```

```
var BEAM_Y = 0.65;
var BEAM_W = 0.8;
var BEAM_H = 0.15;

// scale of the beam, in pixels per meter
var SCALE = 0;

// add a load
function add_load(options, pos, len, mag, handlers)
{
  OOPS.action("insert " + options.type + " load",
      function (arr)
      {
        if (typeof arr == "undefined") // TODO: is redo correctly passing the things?
        {
          var o = options;
          o.mode = ""; // clear the "mini"
          o.start = o.position = pos; // use the dropped X position
          if (o.type.indexOf("distributed") == 0)
          {
            o.length = or(len, 100/SCALE); // default starting length
          }
          else
          {
            o.length = 0;
          }
          o.mag = or(mag, 1);

          console.log(o);
        }
        else
        {
          var o = arr[0], id = arr[1];
          console.log(o);
        }

        var load =  input.load(o, handlers);

        if (typeof id != "undefined")
        {
          Raphael._sets[id] = load;
          load.id = id;
        }

        beam.push( // glom onto the beam
            load
            );

        return load.id;
      },
      function (id)
      {
        var load = input.setById(id);
        load.parent().exclude(load);
        load.remove();

        return [load._o, load.id];
      });
}

function delete_load(load)
```

```
92  {
93    OOPS.action("delete " + load.type + " load",
94      function (loadid)
95      {
96        var dead = input.setById(loadid), b = dead.parent();
97        b.exclude(dead);
98        dead.remove();
99        return [dead._o, dead._d, loadid];
100     },
101     function (arr)
102     {
103       var o = arr[0], d = arr[1], id = arr[2];
104
105       var load = input.load(o, d);
106
107       Raphael._sets[id] = load;
108       load.id = id;
109
110       beam.push(load);
111
112       return id;
113     },
114     load.id);
115 }
116
117 // draws all the beam graphics, given a JSON
118 // representation of the beam and some click/drag handlers
119 function draw_beam(json, s_handlers, l_handlers)
120 {
121   // misc. properties
122   beam.len(json.length);
123   beam.data("mode", json.mode);
124
125   // cantilever support
126   beam.push(input.wall());
127
128   // supports and loads from JSON
129   if (typeof(json.loads) != "undefined")
130   {
131     for (var s in json.supports)
132     {
133       if (json.supports.hasOwnProperty(s))
134       {
135         beam.push(input.support(json.supports[s], s_handlers));
136       }
137     }
138     for (var l in json.loads)
139     {
140       if (json.loads.hasOwnProperty(l))
141       {
142         add_load(json.loads[l],
143                  or(json.loads[l].position, json.loads[l].start),
144                  json.loads[l].end - json.loads[l].start,
145                  json.loads[l].magnitude,
146                  l_handlers);
147       }
148     }
149   }
150 }
151
152 // make the URL up to date, from the current DOM state
```

```
153 function update_url()
154 {
155   history.pushState(null, '', '?' + encodeURIComponent(JSON.stringify(beam.encode())));
156 }
157
158 // switch to cantilever mode
159 function cantilever()
160 {
161   for (var i = 0; i < beam.length; ++i)
162   {
163     if (typeof beam[i].type != "undefined")
164     {
165       if (beam[i].type.indexOf("support") == 0)
166       {
167         beam[i].hide();
168       }
169       else if (beam[i].type == "wall")
170       {
171         beam[i].show();
172       }
173     }
174   }
175
176   draw();
177 }
178
179 // switch to overhanging mode
180 function overhanging()
181 {
182   for (var i = 0; i < beam.length; ++i)
183   {
184     if (typeof beam[i].type != "undefined")
185     {
186       if (beam[i].type.indexOf("support") == 0)
187       {
188         beam[i].show();
189       }
190       else if (beam[i].type == "wall")
191       {
192         beam[i].hide();
193       }
194     }
195   }
196
197   draw();
198 }
199
200 // on startup, create the main Raphael object and the beam (with two supports)
201 window.onload = function () {
202   input_div = document.getElementById("input");
203   input = Raphael(input_div);
204
205   output_div = document.getElementById("output");
206   output = Raphael(output_div);
207
208   EM = getEmSize(document.getElementById("input"))/2;
209   BEAM_H = 4.5*EM/input.height;
210
211   // buttons
212   var buttons = input.set(), undo = input.set(), redo = input.set();
213   undo.push(
```

```
214         input.path("m" + ((input.width*7/10)+40) + "," + 30
215                 + "a 20,20 0,1,0 -20,20")
216           .attr({"arrow-end": "block-wide-long",
217                 "fill-opacity": 0,
218                 "opacity": 0.25,
219                 "stroke-width": 2}),
220         input.text(input.width*7/10 + 20, 30, "UNDO"),
221         input.circle(input.width*7/10 + 20, 30, 20)
222           .attr({"opacity": 0,
223                 "fill": "black"})
224           .click(function () { OOPS.undo(); })
225         );
226     redo.push(
227         input.path("m" + (input.width*8/10) + "," + 30
228                 + "a 20,20 0,1,1 20,20")
229           .attr({"arrow-end": "block-wide-long",
230                 "fill-opacity": 0,
231                 "opacity": 0.25,
232                 "stroke-width": 2}),
233         input.text(input.width*8/10 + 20, 30, "REDO"),
234         input.circle(input.width*8/10 + 20, 30, 20)
235           .attr({"opacity": 0,
236                 "fill": "black"})
237           .click(function () { OOPS.redo(); })
238         );
239     buttons.push(undo, redo);
240
241     OOPS.settings.onChange = function ()
242     {
243       // anything that affects the beam state is undoable,
244       // so anytime the undo state changes is a perfect
245       // time to redraw the graphs
246       setTimeout("draw();", 1); // asynchronous
247
248       // see if we have to enable/disable the undo/redo buttons
249       if (OOPS.undo_queue.length > 0)
250       {
251         undo[0].attr("opacity", 1);
252       }
253       else
254       {
255         undo[0].attr("opacity", 0.25);
256       }
257
258       if (OOPS.redo_queue.length > 0)
259       {
260         redo[0].attr("opacity", 1);
261       }
262       else
263       {
264         redo[0].attr("opacity", 0.25);
265       }
266
267       update_url();
268     }
269
270     // set up the input workspace
271     beam = input.set();
272     toolbox = input.set();
273     var thebeam = input.rect(input.width   *BEAM_X,
274                              input.height  *BEAM_Y,
```

```
                            input.width   *BEAM_W,
                            input.height  *BEAM_H);
    thebeam.type = "beam";

    // wrapper for beam.data("length")
    beam.len = function ()
      {
        return this.data("length", arguments[0]);
      };

    // redraw the beam
    beam.redraw = function ()
      {
        // right now, the DOM is up to date
        //            the graphics might be out of date
        //            the URL might be out of date

        // bring the URL up to speed
        update_url();

        // get rid of all the graphics (and the DOM)
        for (var i = 0; i < beam.length; ++i)
        {
          if (   beam[i].type.indexOf("load") == 0
              || beam[i].type.indexOf("support") == 0
              || beam[i].type == "wall")
          {
            var dead = input.setById(beam[i].id);
            beam.exclude(dead);
            dead.remove();
            --i;
          }
        }

        // redraw using the URL as a source
        draw_beam(JSON.parse(decodeURIComponent(location.search.substring(1))));
      };

    SCALE = input.width*BEAM_W; // px/m

    /* These are generic drag handler factory functions, for doing stuff
     * in the x direction. They take a drawing function (i.e. input.load or input.support)
     * and an update function (to apply dx to the drawing function's options dict) and
     * they handle the drawing of a "ghost" object during the drag operation and the
     * element replacement afterwards.
     *
     * All of the loads and supports are created with invisible drag boxes, so the drag handlers
     * are installed on those and this.parent() is the load/support.
     *
     * Additionally, the drawing functions save their parameters (o and d) on the returned object
     * as _o and _d, so it is easy to recover the parameters for easy modification.
     */
    var move = function (creat, upd) { return function (dx, dy)
        {
          var o = deepcopy(this.parent()._o);

          oarr = upd(o, dx);
          o = oarr[0];
          dx = oarr[1];

          o.mode = "ghost";
```

```
336
337           this.data("ghost").remove();
338           this.data("ghost", creat(o));
339
340           this.transform("t" + dx + ",0");
341           this.dx = dx;
342         }},
343       start = function (creat) { return function ()
344         {
345           this.dx = 0;
346           this.parent().hide();
347           this.data("ghost", creat(this.parent()._o));
348         }},
349       end = function (creat, upd) { return function ()
350         {
351           function replace(leave, come)
352           {
353             var id = leave.id, guardian = leave.parent();
354
355             guardian.exclude(leave);
356             leave.remove();
357
358             // even though we are ghosting,
359             // make sure the load has a stable id (for undo/redo)
360             come.id = id;
361             Raphael._sets[id] = come; // HACK
362             guardian.push(come);
363             console.log(id);
364
365             return id;
366           }
367
368           var options = deepcopy(this.parent()._o), dx = this.dx, box = this;
369
370           this.data("ghost").remove();
371           this.transform("");
372
373           OOPS.action("move/resize " + options.type + " load by dx=" + dx,
374               function (arr)
375               {
376                 var dx = arr[0], upd = arr[1][0], creat = arr[1][1],
377                   load = input.setById(arr[2]),
378                   o = deepcopy(load._o);
379
380                 o = upd(o, dx)[0];
381
382                 return [dx, [upd, creat], replace(load, creat(o, load._d))];
383               },
384               function (arr)
385               {
386                 var dx = arr[0], upd = arr[1][0], creat = arr[1][1],
387                   load = input.setById(arr[2]),
388                   o = deepcopy(load._o);
389
390                 o = upd(o, -dx)[0];
391
392                 return [dx, [upd, creat], replace(load, creat(o, load._d))];
393               },
394               [dx, [upd, creat], this.parent().id]);
395         }};
396
```

```
397     // here we make different update functions for move, left-resize and right-resize
398     var m_upd = function (o, dx)
399           {
400             o.position += dx/SCALE*beam.len();
401             if (o.position < 0)
402             {
403               dx -= o.position*SCALE/beam.len();
404               o.position = 0;
405             }
406             if (o.length)
407             {
408               if (o.position + o.length > (input.width*BEAM_W/SCALE*beam.len()))
409               {
410                 dx -= (o.position + o.length)*SCALE/beam.len() - (input.width*BEAM_W);
411                 o.position = input.width*BEAM_W/SCALE*beam.len() - o.length;
412               }
413             }
414             else
415             {
416               if (o.position > (input.width*BEAM_W/SCALE*beam.len()))
417               {
418                 dx -= o.position*SCALE/beam.len() - input.width*BEAM_W;
419                 o.position = input.width*BEAM_W/SCALE*beam.len();
420               }
421             }
422             return [o, dx];
423           },
424       left_upd = function (o, dx)
425            {
426              o.position += dx/SCALE*beam.len();
427              o.length -= dx/SCALE*beam.len();
428              if (o.position < 0)
429              {
430                dx -= o.position*SCALE/beam.len();
431                o.length += o.position;
432                o.position = 0;
433              }
434              if (o.length <= 0)
435              {
436                dx += o.length*SCALE/beam.len();
437                o.position += o.length;
438                o.length = 0;
439              }
440              return [o, dx];
441            },
442       right_upd = function (o, dx)
443             {
444               o.length += dx/SCALE*beam.len();
445               if (o.position + o.length > (input.width*BEAM_W/SCALE*beam.len()))
446               {
447                 dx -= (o.position + o.length)*SCALE/beam.len() - (input.width*BEAM_W);
448                 o.length -= (o.position + o.length)
449                       - (input.width*BEAM_W/SCALE*beam.len());
450               }
451               if (o.length <= 0)
452               {
453                 dx -= o.length*SCALE/beam.len();
454                 o.length = 0;
455               }
456               return [o, dx];
457             },
```

```
458
459       // now use the factories to make drag handlers
460       l_move = move(input.load, m_upd),
461       left_move = move(input.load, left_upd),
462       right_move = move(input.load, right_upd),
463       s_move = move(input.support, m_upd),
464
465       l_start = start(input.load),
466       left_start = start(input.load),
467       right_start = start(input.load),
468       s_start = start(input.support),
469
470       l_end = end(input.load, m_upd),
471       left_end = end(input.load, left_upd),
472       right_end = end(input.load, right_upd),
473       s_end = end(input.support, m_upd),
474
475       /* the drag handlers for the toolbox items are still special-cased
476        * since they need to move in both the x and y directions
477        * and they don't stay where they are dropped (they either disappear or
478        * snap on to the beam)
479        */
480       t_move = function (dx, dy)
481         {
482           this.parent().transform("t" + dx + "," + dy);
483           this.dx = dx;
484           this.dy = dy;
485         },
486       t_start = function ()
487         {
488           toolbox.push(input.load(this.parent()._o, this.parent()._d));
489           this.ox = this.attr("x");
490           this.oy = this.attr("y");
491         },
492       t_end = function ()
493         {
494           // is it close to the beam?
495           if ((this.ox + this.dx) >= input.width*BEAM_X
496            && (this.ox + this.dx) <= input.width*(BEAM_X + BEAM_W)
497            && (this.oy + this.dy) >= input.height*(BEAM_Y - 2*BEAM_H)
498            && (this.oy + this.dy) <= input.height*(BEAM_Y + BEAM_H))
499           {
500             // add it!
501             var options = deepcopy(this.parent()._o), ox = this.ox, dx = this.dx;
502
503             add_load(options,
504                      (ox + dx - input.width*BEAM_X)/SCALE*beam.len(),
505                      undefined, undefined,
506                      [l_move, l_start, l_end,
507                       left_move, l_start, left_end,
508                       right_move, l_start, right_end]);
509           }
510
511           this.parent().remove();
512         };
513
514     thebeam.attr("fill", "#BBB");
515     thebeam.attr("stroke", "#666"); // a positively DEVILISH shade of gray
516
517     toolbox.push( // put all the stuff in the toolbox
518         input.text(100, 25, "Add loads:").scale(1.5, 1.5),
```

```
519         input.load(
520           {type: "concentrated", resizeable: false, mode: "mini",
521            x: 170, y: 30, w: 0, h: 15},
522           [t_move, t_start, t_end]),
523         input.load(
524           {type: "moment", resizeable: false, mode: "mini",
525            x: 220, y: 30, w: 15, h: 15},
526           [t_move, t_start, t_end]),
527         input.load(
528           {type: "distributed-constant", mode: "mini",
529            x: 270, y: 30, w: 50, h: 15},
530           [t_move, t_start, t_end]),
531         input.load(
532           {type: "distributed-linear", mode: "mini",
533            x: 365, y: 30, w: 50, h: 15},
534           [t_move, t_start, t_end])
535         );
536
537     beam.data("length", 1);
538     beam.data("mode", "overhanging");
539     beam.push(
540         thebeam
541         );
542
543     // if the URL has a beam encoded in it, load it up
544     if (location.search != "")
545     {
546       var json = JSON.parse(decodeURIComponent(location.search.substring(1)));
547
548       draw_beam(json, [s_move, s_start, s_end], [l_move, l_start, l_end,
549                                                  left_move, l_start, left_end,
550                                                  right_move, l_start, right_end]);
551     }
552     else
553     {
554       beam.push( // create the default set of supports for a
555                  // simply supported (possibly overhanging) beam
556           input.support({type: "pinned", position: 0}, [s_move, s_start, s_end]),
557           input.support({type: "roller", position: 1}, [s_move, s_start, s_end]),
558           input.wall()
559           );
560     }
561
562     // clicklabel for resizing the beam / changing the mode
563     beam.push(
564         input.text(input.width*(BEAM_X*0.85),
565                input.height*(BEAM_Y + BEAM_H/2),
566                beam.data("mode"))
567           .attr("text-anchor", "end")
568           .click(function ()
569                {
570                  var t = this;
571                  function switch_mode()
572                  {
573                    switch (beam.data("mode"))
574                    {
575                      case "overhanging":
576                        beam.data("mode", "cantilever");
577                        cantilever();
578                        break;
579                      case "cantilever":
```

```
                    beam.data("mode", "overhanging");
                    overhanging();
                    break;
                  default:
                    break;
                }
                t.attr("text", beam.data("mode"));
              }

              OOPS.action("switch beam mode",
                          switch_mode,
                          switch_mode);
            }),
        input.text(input.width*(BEAM_X + BEAM_W*1.05),
                   input.height*(BEAM_Y + BEAM_H/2),
                   beam.len() + " m")
          .attr("text-anchor", "start")
          .click(function ()
              {
                var str = prompt("New value for beam length (m):"
                       + "\n\nsyntax: ST#"
                       + "\n\tS is where to anchor the resize (L left, R right, M middle)"
                       + "\n\tT is whether to keep absolute load/support positions (+)"
                         + "or scale them with beam length (=)"
                       + "\n\t# is the new beam length\n\n", "L+" + beam.len());
                if (str !== null && str.length >= 3)
                {
                  var S = str[0], T = str[1], val = parseFloat(str.slice(2));
                  if ("LRM".indexOf(S) != -1 && "=+".indexOf(T) != -1 && !isNaN(val))
                  {
                    var t = this;
                    OOPS.action("change beam length from " + beam.len() + " to " + val + " m",
                      function (arr)
                      {
                        var S = arr[0], T = arr[1], len = arr[2];
                        var old_len = beam.len();
                        beam.len(len);
                        if (T == "=")
                        {
                          for (var i = 0; i < beam.length; ++i)
                          {
                            if (beam[i].type.indexOf("load") == 0
                                || beam[i].type.indexOf("support") == 0)
                            {
                              if (typeof beam[i]._data.position != "undefined")
                                beam[i]._data.position *= len / old_len;
                              if (typeof beam[i]._data.start    != "undefined")
                                beam[i]._data.start    *= len / old_len;
                              if (typeof beam[i]._data.end      != "undefined")
                                beam[i]._data.end      *= len / old_len;
                              if (typeof beam[i]._data.length   != "undefined")
                                beam[i]._data.length   *= len / old_len;
                            }
                          }
                        }
                        else // T = "+"
                        {
                          for (var i = 0; i < beam.length; ++i)
                          {
                            if (beam[i].type.indexOf("load") == 0
                                || beam[i].type.indexOf("support") == 0)
```

```
{
  function check_outlier(i)
  {
    if (   (typeof beam[i]._data.position != "undefined"
            && (beam[i]._data.position < 0
                || beam[i]._data.position > len))
        || (typeof beam[i]._data.start    != "undefined"
            && (beam[i]._data.start < 0
                || beam[i]._data.start > len))
        || (typeof beam[i]._data.end      != "undefined"
             && (beam[i]._data.end < 0
                || beam[i]._data.end > len)))
    {
      if (beam[i].type.indexOf("support") == 0)
      {
        beam[i]._data.position = len;
      }
      else
      {
        // not undoable?
        var dead = beam[i];
        beam.exclude(beam[i]);
        dead.remove();
        return i - 1; // HACK
      }
    }
    return i;
  }

  if (S == "R")
  {
    // move loads
    if (typeof beam[i]._data.position != "undefined")
      beam[i]._data.position = (len
                                - (old_len - beam[i]._data.position));

    if (typeof beam[i]._data.start    != "undefined")
      beam[i]._data.start    = (len
                                - (old_len - beam[i]._data.start));

    if (typeof beam[i]._data.end      != "undefined")
      beam[i]._data.end      = (len
                                - (old_len - beam[i]._data.end));
  }
  else if (S == "M")
  {
    // move loads
    if (typeof beam[i]._data.position != "undefined")
      beam[i]._data.position = (len/2
                                + (beam[i]._data.position - old_len/2));

    if (typeof beam[i]._data.start    != "undefined")
      beam[i]._data.start    = (len/2
                                + (beam[i]._data.start - old_len/2));

    if (typeof beam[i]._data.end      != "undefined")
      beam[i]._data.end      = (len/2
                                + (beam[i]._data.end - old_len/2));
  }
  i = check_outlier(i);
}
```

```
                        }
                      }
                      t.attr("text", len + " m");
                      beam.redraw();
                      return [S, T, old_len];
                    },
                    function (arr)
                    {
                      var S = arr[0], T = arr[1], len = arr[2];
                      var old_len = beam.len();
                      beam.len(len);
                      t.attr("text", len + " m");
                      beam.redraw();
                      return [S, T, old_len];
                    },
                    [S, T, val]);
                }
              }
            })
      );
}
```

6.5 graphics.js

```
/* Diagramr special Raphael drawing functions
 *
 * Alex Burka, October 2011
 */

// measure font height
// http://eriwen.com/?p=40
function getEmSize(el)
{
  return Number(getComputedStyle(el, '').fontSize.match(/(\d+)px/)[1]);
}

// round to specified number of decimal places
function round(n, d)
{
  var factor = Math.pow(10, d);
  return Math.round(n*factor)/factor;
}

// move a load/support, after updating its options dict
// parameters:
//  - thing: a child of the load/support set
//  - creat: the function to create a new one (input.load or input.support)
//  - upd: function to update the options dict
function redraw(thing, creat, upd)
{
  var l = thing.parent(), b = l.parent(), o = upd(deepcopy(l._o));
  var n = creat(o, l._d);
  n.id = l.id;
  Raphael._sets[l.id] = n;
  b.push(n);
  b.exclude(l);
  l.remove();
  return n;
}

// draw the wall for a cantilever beam
```

```
Raphael.fn.wall = function () {
  var x = input.width*(BEAM_X + BEAM_W),
    y = input.height*(BEAM_Y - BEAM_H/2),
    w = input.width*(0.1*BEAM_X),
    h = input.height*(2*BEAM_H);

  var wall = input.set();
  wall.push(
      input.rect(x, y, w/2, h)
        .attr("fill", "blue"),

      input.path("m" + (x+w/2) + "," + y

               + "l" + (w/2)  + "," + (h/10)

               + "m" + -w/2   + "," + 0
               + "l" + (w/2)  + "," + (h/10)

               + "m" + -w/2   + "," + 0
               + "l" + (w/2)  + "," + (h/10)

               + "m" + -w/2   + "," + 0
               + "l" + (w/2)  + "," + (h/10)

               + "m" + -w/2   + "," + 0
               + "l" + (w/2)  + "," + (h/10)

               + "m" + -w/2   + "," + 0
               + "l" + (w/2)  + "," + (h/10)

               + "m" + -w/2   + "," + 0
               + "l" + (w/2)  + "," + (h/10)

               + "m" + -w/2   + "," + 0
               + "l" + (w/2)  + "," + (h/10)

               + "m" + -w/2   + "," + 0
               + "l" + (w/2)  + "," + (h/10)

               + "m" + -w/2   + "," + 0
               + "l" + (w/2)  + "," + (h/10))
      );

  if (beam.data("mode") == "overhanging")
  {
    wall.hide();
  }

  wall.type = "wall";
  return wall;
};

// draw a beam support
//  parameters:
//    - o (required): options dict
//      type (required): support type. currently "pinned" or "roller"
//      position (required): position along the beam, in meters
//    - d (optional): drag handler array
//      d[0] = onmove, d[1] = onstart, d[2] = onend
//  returns: support element
//    type: support_{o.type}
```

```
99  //    data: "position", in meters
100 Raphael.fn.support = function (o, d) {
101   var supp = input.set();
102   supp._o = o;
103   supp._d = d;
104
105   var type = o.type, position = o.position;
106
107   var x = position*SCALE/beam.len() + input.width*BEAM_X;
108   var y = input.height*(BEAM_Y + BEAM_H);
109   var w = input.width*BEAM_W / 20;
110   var h = input.height*BEAM_H * 2/3;
111   var r = w/10;
112
113   if (typeof d != "undefined")
114   {
115     supp.push(
116         input.rect(x-w/2, y, w, h)
117             .attr("fill", "black")
118             .attr("opacity", 0)
119             .attr("cursor", "move")
120             .drag(d[0], d[1], d[2])
121         );
122   }
123
124   supp.push(
125       input.path("m" + x     + "," + y
126                + "l" + w/2 + "," + (type == "roller" ? h-2*r : h)
127                + " " + (-w) + "," + 0
128                + "z")
129       );
130
131   y += (type == "roller" ? h-2*r : h);
132   x -= w/2;
133
134   if (type == "roller")
135   {
136     supp.push(
137         input.circle(x + 2*r, y+r, r),
138         input.circle(x + 5*r, y+r, r),
139         input.circle(x + 8*r, y+r, r),
140         input.path("m" + x   + "," + (y+2*r)
141                  + "h" + w)
142         );
143
144     y += 2*r;
145   }
146
147   supp.push(
148       input.path("m" + x    + "," + y
149                + "l" + (-r)   + "," + r
150
151                + "m" + (3*r)  + "," + (-r)
152                + "l" + (-r)  + "," + r
153
154                + "m" + (3*r)  + "," + (-r)
155                + "l" + (-r)  + "," + r
156
157                + "m" + (3*r)  + "," + (-r)
158                + "l" + (-r)  + "," + r
159
```

```
                + "m" + (3*r)  + "," + (-r)
                + "l" + (-r)  + "," + r

                + "m" + (3*r)  + "," + (-r)
                + "l" + (-r)  + "," + r)
        );

    supp.push(
        input.text(x + w/2, y + r + EM, round(position,3) + " m").click(function ()
          {
            var val = parseFloat(prompt("New value for position (m):", o.position));
            if (!isNaN(val))
            {
              var type = this.parent()._o.type, from = this.parent()._o.position, to = val;
              OOPS.action("move " + type + " support from x=" + from + " to x=" + to,
                function (supp)
                {
                  return redraw(supp[0],
                                input.support,
                                function (o)
                                {
                                  o.position = to; return o;
                                });
                },
                function (supp)
                {
                  return redraw(supp[0],
                                input.support,
                                function (o)
                                {
                                  o.position = from; return o;
                                });
                },
                this.parent());
            }
          })
        );

    supp.attr("fill", "blue");
    supp.type = "support_" + type;
    supp.data("position", position);
    supp[0].toFront(); // drag box above

    if (beam.data("mode") == "cantilever")
    {
      supp.hide();
    }

    return supp;
  };

  // draw a beam load
  //  parameters:
  //    - o (required): options dict
  //      type (required): load type. currently "concentrated",
  //                                            "distributed-constant",
  //                                            "distributed-linear"
  //        concentrated: o.position
  //        distributed-*: o.start, o.length
  //        all types: o.mag (not required in mini mode)
  //      mode (optional): affects the drawing
```

```
221 //        "mini": use o.x/o.y/o.w/o.h to draw the support in the place specified
222 //        "ghost": draw normally, but with greater arrow spacing (much faster)
223 //        default: draw normally
224 //    - d (optional): drag handler array
225 //      for just move: d[0] = onmove, d[1] = onstart, d[2] = onend
226 //      for resize as well: d[3..5] for left, d[6..8] for right
227 //  returns: support element
228 //    type: support_{o.type}
229 //    data: "position", in meters
230 Raphael.fn.load = function (o, d)
231 {
232   /* install drag box and handler functions
233    * o: load parameters
234    * d: drag handler funcs
235    * x/y/w/h: position of the load
236    */
237   function install_handles(o, d, x, y, w, h)
238   {
239     switch (d.length)
240     {
241       case 3:
242         load.push(input.rect(x, y-h, w, h)
243               .attr({
244                 opacity: 0,
245                 fill: "black",
246                 cursor: o.mode == "mini" ? "s-resize" : "move"})
247               .drag(d[0], d[1], d[2])
248              );
249         break;
250       case 9:
251         if (o.resizeable == false)
252         {
253           return install_handles(o, d.slice(0, 3), x, y, w, h);
254         }
255         load.push(input.rect(x, y-h, w, h)
256               .attr({
257                 opacity: 0,
258                 fill: "black",
259                 cursor: o.mode == "mini" ? "s-resize" : "move"})
260               .drag(d[0], d[1], d[2]),
261               input.rect(x, y-h, w/5, h)
262               .attr({
263                 opacity: 0,
264                 fill: "black",
265                 cursor: "w-resize"})
266               .drag(d[3], d[4], d[5]),
267               input.rect(x+w*4/5, y-h, w/5, h)
268               .attr({
269                 opacity: 0,
270                 fill: "black",
271                 cursor: "e-resize"})
272               .drag(d[6], d[7], d[8])
273              );
274         break;
275       default:
276         console.log("weird number of drag handlers");
277         break;
278     }
279   }
280
281   function distributed(start, end, length, y, h, units)
```

```
282     {
283       load.data("start", (start - input.width*BEAM_X)/SCALE*beam.len());
284       load.data("end", (end - input.width*BEAM_X)/SCALE*beam.len());
285       load.push(
286           input.text(start + length/2, y - h - EM, o.mag + " " + units).click(function ()
287             {
288               var val = parseFloat(prompt("New value for magnitude (" + units + "):", o.mag));
289               if (!isNaN(val))
290               {
291                 var type = this.parent()._o.type, from = this.parent()._o.mag, to = val;
292                 if (val == 0)
293                 {
294                   delete_load(this.parent());
295                 }
296                 else
297                 {
298                   OOPS.action("change magnitude of " + type + " load from " + from + " to " + to,
299                     function (loadid)
300                     {
301                       return redraw(input.setById(loadid)[0],
302                                     input.load,
303                                     function (o)
304                                     {
305                                       o.mag = to;
306                                       return o;
307                                     }).id;
308                     },
309                     function (loadid)
310                     {
311                       return redraw(input.setById(loadid)[0],
312                                     input.load,
313                                     function (o)
314                                     {
315                                       o.mag = from;
316                                       return o;
317                                     }).id;
318                     },
319                     this.parent().id);
320                 }
321               }
322             }),
323           input.text(start, y + EM, round(o.position, 2) + " m >").click(function ()
324             {
325               var str = prompt("New value for start (m): (prepend '=' to keep length)", o.position);
326               if (str !== null && str.length != 0)
327               {
328                 if (str.charAt(0) == '=')
329                 {
330                   var val = parseFloat(str.substr(1));
331                   if (!isNaN(val))
332                   {
333                     var type = this.parent()._o.type, from = this.parent()._o.position, to = val;
334                     OOPS.action("move left side of " + o.type + " load from x="
335                                 + from + " to x=" + to + " (preserving length)",
336                       function (loadid)
337                       {
338                         return redraw(input.setById(loadid)[0],
339                                       input.load,
340                                       function (o)
341                                       {
342                                         o.position = to;
```

```
                                    return o;
                                  }).id;
                    },
                    function (loadid)
                    {
                      return redraw(input.setById(loadid)[0],
                                    input.load,
                                    function (o)
                                    {
                                      o.position = from;
                                      return o;
                                    }).id;
                    },
                    this.parent().id);
                }
              }
              else
              {
                var val = parseFloat(str);
                if (!isNaN(val))
                {
                  var type = this.parent()._o.type,
                      frompos = this.parent()._o.position,
                      fromlen = this.parent()._o.length,
                      to = val;
                  OOPS.action("move left side of " + type + " load from x=" + frompos
                              + " to x=" + to + " (resizing)",
                      function (loadid)
                      {
                        return redraw(input.setById(loadid)[0],
                                      input.load,
                                      function (o)
                                      {
                                        o.length += o.position - to;
                                        o.position = to;
                                        return o;
                                      }).id;
                      },
                      function (loadid)
                      {
                        return redraw(input.setById(loadid)[0],
                                      input.load,
                                      function (o)
                                      {
                                        o.length = fromlen;
                                        o.position = frompos;
                                        return o;
                                      }).id;
                      },
                      this.parent().id);
                }
              }
            }
          }).attr("text-anchor", "start"),
        input.text(start + length/2, y + EM*2.5, "<|  " + round(o.length, 2) + " m  |>")
          .click(function ()
          {
            var val = parseFloat(prompt("New value for length (m):", o.length));
            if (!isNaN(val))
            {
              var type = this.parent()._o.type, from = this.parent()._o.length, to = val;
```

```
404               OOPS.action("resize " + type + " load from L=" + from + " to L=" + to,
405                 function (loadid)
406                 {
407                   return redraw(input.setById(loadid)[0],
408                                 input.load,
409                                 function (o)
410                                 {
411                                   o.length = to;
412                                   return o;
413                                 }).id;
414                 },
415                 function (loadid)
416                 {
417                   return redraw(input.setById(loadid)[0],
418                                 input.load,
419                                 function (o)
420                                 {
421                                   o.length = from;
422                                   return o;
423                                 }).id;
424                 },
425                 this.parent().id);
426             }
427           }),
428         input.text(start + length, y + EM*4, "< " + round(o.position + o.length, 2) + " m")
429           .click(function ()
430           {
431             var str = prompt("New value for end (m): (prepend '=' to keep length)",
432                              o.position + o.length);
433             if (str !== null && str.length != 0)
434             {
435               if (str.charAt(0) == '=')
436               {
437                 var val = parseFloat(str.substr(1));
438                 if (!isNaN(val))
439                 {
440                   var type = this.parent()._o.type, from = this.parent()._o.position;
441                   OOPS.action("move end of " + type + " load by dx=" + val
442                               + " (length-preserving)",
443                     function (loadid)
444                     {
445                       return redraw(input.setById(loadid)[0],
446                                     input.load,
447                                     function (o)
448                                     {
449                                       o.position = val -  o.length; return o;
450                                     }).id;
451                     },
452                     function (loadid)
453                     {
454                       return redraw(input.setById(loadid)[0],
455                                     input.load,
456                                     function (o)
457                                     {
458                                       o.position = from;
459                                       return o;
460                                     }).id;
461                     },
462                     this.parent().id);
463                 }
464               }
```

```
465                 else
466                 {
467                   var val = parseFloat(str);
468                   if (!isNaN(val))
469                   {
470                     var type = this.parent()._o.type, from = this.parent()._o.length;
471                     OOPS.action("move end of " + type + " load by dx=" + val + " (resizing)",
472                         function (loadid)
473                         {
474                           return redraw(input.setById(loadid)[0],
475                                         input.load,
476                                         function (o)
477                                         {
478                                           o.length = val -  o.position;
479                                           return o;
480                                         }).id;
481                         },
482                         function (loadid)
483                         {
484                           return redraw(input.setById(loadid)[0],
485                                         input.load,
486                                         function (o)
487                                         {
488                                           o.length = from;
489                                           return o;
490                                         }).id;
491                         },
492                         this.parent().id);
493                   }
494                 }
495               }
496             }).attr("text-anchor", "end")
497         );
498     }
499
500     var load = input.set();
501     load._o = o;
502     load._d = d;
503
504     switch (o.type)
505     {
506       case "concentrated":
507         var x = o.mode == "mini" ? o.x : input.width*BEAM_X + o.position*SCALE/beam.len();
508         var h = o.mode == "mini" ? o.h : input.height*BEAM_H*3/2;
509         var y = o.mode == "mini" ? o.y : input.height*BEAM_Y - h/20;
510         var w = h*2/9;
511         load.push(
512             input.arrow(x, y - h, x, y)
513             );
514         if (o.mode != "mini")
515         {
516           load.data("position", o.position);
517           load.push(
518               input.text(x, y - h - EM, o.mag + " N").click(function ()
519                 {
520                   var val = parseFloat(prompt("New value for magnitude (N):", o.mag));
521                   if (!isNaN(val))
522                   {
523                     var type = this.parent()._o.type, from = this.parent()._o.mag;
524                     if (val == 0)
525                     {
```

```
526                     delete_load(this.parent());
527                   }
528                   else
529                   {
530                     OOPS.action("change magnitude of " + type + " load from "
531                                 + from + " to " + val,
532                       function (loadid)
533                       {
534                         return redraw(input.setById(loadid)[0],
535                                       input.load,
536                                       function (o)
537                                       {
538                                         o.mag = val;
539                                         return o;
540                                       }).id;
541                       },
542                       function (loadid)
543                       {
544                         return redraw(input.setById(loadid)[0],
545                                       input.load,
546                                       function (o)
547                                       {
548                                         o.mag = from;
549                                         return o;
550                                       }).id;
551                       },
552                       this.parent().id);
553                   }
554                 }
555               }),
556             input.text(x, y + EM, round(o.position, 3) + " m").click(function ()
557               {
558                 var val = parseFloat(prompt("New value for position (m):", o.position));
559                 if (!isNaN(val))
560                 {
561                   var type = this.parent()._o.type, from = this.parent()._o.position;
562                   OOPS.action("move " + type + " load from x=" + from + " to x=" + val,
563                     function (loadid)
564                     {
565                       return redraw(input.setById(loadid)[0],
566                                     input.load,
567                                     function (o)
568                                     {
569                                       o.position = val;
570                                       return o;
571                                     }).id;
572                     },
573                     function (loadid)
574                     {
575                       return redraw(input.setById(loadid)[0],
576                                     input.load,
577                                     function (o)
578                                     {
579                                       o.position = from;
580                                       return o;
581                                     }).id;
582                     },
583                     this.parent().id);
584                 }
585               })
586             );
```

```
587         }
588         if (typeof d != "undefined")
589         {
590           install_handles(o, d, x-w*2, y, w*4, h);
591         }
592         break;
593       case "moment":
594         var x = o.mode == "mini" ? o.x   : input.width*BEAM_X + o.position*SCALE/beam.len();
595         var h = o.mode == "mini" ? o.h   : input.height*BEAM_H*4/5;
596         var w = h;
597         var y = o.mode == "mini" ? o.y-h/2 : input.height*BEAM_Y + input.height*BEAM_H/2;
598
599         if (o.mode == "mini" || o.mag >= 0)
600         {
601           load.push(
602               input.path("m" + (x+w/2) + "," + y
603                        + "a" + (w/2) + "," + (h/2) + ",0,1,0," + (-w/2) + "," + (h/2))
604                 .attr("arrow-end", "block-wide-long")
605                 .attr("fill-opacity", "0"),
606               input.circle(x, y, w/15).attr("fill", "black")
607               );
608         }
609         else
610         {
611           // curl the other way if negative
612           load.push(
613               input.path("m" + (x-w/2) + "," + y
614                        + "a" + (w/2) + "," + (h/2) + ",0,1,1," + (w/2) + "," + (h/2))
615                 .attr("arrow-end", "block-wide-long")
616                 .attr("fill-opacity", "0"),
617               input.circle(x, y, w/15).attr("fill", "black")
618               );
619         }
620
621         if (o.mode != "mini")
622         {
623           load.data("position", o.position);
624           load.push(
625               input.text(x, y - input.height*BEAM_H*2/3, o.mag + " Nm")
626                 .click(function ()
627                 {
628                   var val = parseFloat(prompt("New value for magnitude (N):", o.mag));
629                   if (!isNaN(val))
630                   {
631                     var type = this.parent()._o.type, from = this.parent()._o.mag;
632                     if (val == 0)
633                     {
634                       delete_load(this.parent());
635                     }
636                     else
637                     {
638                       OOPS.action("change magnitude of " + type + " load from "
639                                   + from + " to " + val,
640                         function (loadid)
641                         {
642                           return redraw(input.setById(loadid)[0],
643                                         input.load,
644                                         function (o)
645                                         {
646                                           o.mag = val;
647                                           return o;
```

```
                                        }).id;
                        },
                        function (loadid)
                        {
                          return redraw(input.setById(loadid)[0],
                                        input.load,
                                        function (o)
                                        {
                                          o.mag = from;
                                          return o;
                                        }).id;
                        },
                        this.parent().id);
                    }
                  }
                }),
              input.text(x, y + input.height*BEAM_H*2/3, round(o.position, 3) + " m")
                .click(function ()
                {
                  var val = parseFloat(prompt("New value for position (m):", o.position));
                  if (!isNaN(val))
                  {
                    var type = this.parent()._o.type, from = this.parent()._o.position;
                    OOPS.action("move " + type + " load from x=" + from + " to " + val,
                      function (loadid)
                      {
                        return redraw(input.setById(loadid)[0],
                                      input.load,
                                      function (o)
                                      {
                                        o.position = val;
                                        return o;
                                      }).id;
                      },
                      function (loadid)
                      {
                        return redraw(input.setById(loadid)[0],
                                      input.load,
                                      function (o)
                                      {
                                        o.position = from;
                                        return o;
                                      }).id;
                      },
                      this.parent().id);
                  }
                })
              );
          }
          if (typeof d != "undefined")
          {
            install_handles(o, d, x-w/2, y+h/2, w, h);
          }
          break;
        case "distributed-constant":
          var x = o.mode == "mini" ? o.x : input.width*BEAM_X + o.position*SCALE/beam.len();
          var h = o.mode == "mini" ? o.h : input.height*BEAM_H;
          var y = o.mode == "mini" ? o.y : input.height*BEAM_Y - h/20;

          var start = x;
          var end = x + (o.mode == "mini" ? o.w : o.length*SCALE/beam.len());
```

```
709         var length = end - start;
710         var tick = length / Math.ceil(length / (o.mode == "ghost" ? 50 : 15));
711 
712         load.push(
713             input.path("m" + x + "," + (y - h)
714                 + "h" + length)
715             );
716         for (x = start; x < (end+tick/2); x += tick)
717         {
718           load.push(
719               input.arrow(x, y - h, x, y)
720               );
721         }
722 
723         if (o.mode != "mini")
724         {
725           distributed(start, end, length, y, h, "N/m");
726         }
727         if (typeof d != "undefined")
728         {
729           install_handles(o, d, start, y, length, h);
730         }
731         break;
732       case "distributed-linear":
733         var x = o.mode == "mini" ? o.x : input.width*BEAM_X + o.position*SCALE/beam.len();
734         var h = o.mode == "mini" ? o.h : input.height*BEAM_H;
735         var y = o.mode == "mini" ? o.y : input.height*BEAM_Y - h/20;
736 
737         var start = x;
738         var end = x + (o.mode == "mini" ? o.w : o.length*SCALE/beam.len());
739         var length = end - start;
740         var tick = length / Math.ceil(length / (o.mode == "ghost" ? 50 : 15));
741 
742         if (o.mode == "mini" || o.mag >= 0)
743         {
744           load.push(
745               input.path("m" + x + "," + y
746                       + "l" + length + "," + (-h))
747               );
748           for (x = start+tick; x < (end+tick/2); x += tick)
749           {
750             load.push(
751                 input.arrow(x, y - h*(x-start)/length, x, y)
752                 );
753           }
754         }
755         else
756         { // slope the other way if negative
757           load.push(
758               input.path("m" + x + "," + (y-h)
759                   + "l" + length + "," + h)
760               );
761           for (x = start; x < end; x += tick)
762           {
763             load.push(
764                 input.arrow(x, y - h*(end-x)/length, x, y)
765                 );
766           }
767         }
768 
769         if (o.mode != "mini")
```

```
      {
        distributed(start, end, length, y, h, "N/m/m");
      }
      if (typeof d != "undefined")
      {
        install_handles(o, d, start, y, length, h);
      }
      break;
    default:
      console.log("unsupported load type " + o.type);
  }

  load.attr("fill", "black");
  load.type = "load_" + o.type;
  if (o.mode != "mini") load.data("mag", o.mag);
  return load;
};


function draw_singularity(fns, length, reactions, labels)
{
  colors = ["#ff0000", "#00ff00", "#0000ff"];

  function value(fn, x)
  {
    var val = 0;
    for (var term in fn)
    {
      if (fn.hasOwnProperty(term))
      {
        if (x >= fn[term][1] && fn[term][2] >= 0)
        {
          val += fn[term][0] * Math.pow(x - fn[term][1], fn[term][2]);
        }
      }
    }
    return val;
  }

  output.clear();

  var POINTS = 500;
  var x = Array(POINTS+1), y = Array(fns.length), zero = Array(POINTS+1);
  for (var j = 0; j < fns.length; ++j) { y[j] = Array(POINTS+1); }
  for (var i = 0; i <= POINTS; i++)
  {
    x[i] = i*length/POINTS;
    zero[i] = 0;
    for (var j = 0; j < fns.length; ++j)
    {
      y[j][i] = -value(fns[j], x[i]);
    }
  }

  for (var j = 0; j < fns.length; ++j)
  {
    output.linechart(output.width*BEAM_X,
                     output.height*.1 + j*output.height*.8/fns.length,
                     output.width*BEAM_W,
                     output.height*.8 / fns.length,
                     [x, x], [y[j], zero],
```

```
                      {axis:   "0 1 1 1",
                       colors: [colors[j], "black"]});

    output.text(output.width*(BEAM_X*0.4),
                output.height*.3 + j*output.height*.8/fns.length,
                labels[j]);
  }

  if (typeof reactions != "undefined")
  {
    output.text(30, 20, "Ra: " + round(reactions[0], 3)
                      + "\nRb: " + round(reactions[1], 3));
    var s = "M(x) = ";
    for (var term in fns[1])
    {
      if (fns[1].hasOwnProperty(term))
      {
        s += round(-fns[1][term][0], 3)
                    + "<x - " + round(fns[1][term][1], 3) + " >^"
                    + round(fns[1][term][2], 3) + " + ";
      }
    }
    output.text(output.width/2, 20, s.substr(0, s.length - 2));
  }
}
```

6.6 oops.js

```
/* Oops: JavaScript undo/redo library
 *
 * Alex Burka
 * November 2011
 */

OOPS = {
  // START public interface
  settings: { // external settings
    max_undo: 20, // CURRENTLY IGNORED
    onChange: function () {}
  },

  action: function (name, forward, backward, starting_param)
    {
      this.redo_queue = [];
      this.register_undo(name, forward, backward, forward(starting_param));
      this.settings.onChange();
    },

  actionFn: function (name, fn, for_args, back_args)
    {
      return this.action(name,
          function () { return fn.apply(for_args); },
          function () { return fn.apply(back_args); });
    },

  group: function ()
    {
      // TODO: groups within groups
      if (this.group)
      {
        this.undo_queue[this.undo_queue.length-1].name += " }";
```

```
34          this.group = false;
35        }
36        else
37        {
38          this.group = true;
39          this.undo_queue[this.undo_queue.length-1].name += "{ ";
40        }
41      },
42
43    undo: function ()
44      {
45        console.log("OOPS.undo");
46        if (this.group)
47        {
48          throw "no undoing while in a group!";
49        }
50        else if (this.undo_queue.length > 0)
51        {
52          var item = this.undo_queue.pop();
53
54          for (var fn in item.backward)
55          {
56            if (item.backward.hasOwnProperty(fn))
57            {
58              item.thing[fn] = item.backward[fn](item.thing[fn]);
59            }
60          }
61          console.log(item.thing);
62          this.redo_queue.push(item);
63          this.settings.onChange();
64        }
65      },
66
67    redo: function ()
68      {
69        console.log("OOPS.redo");
70        if (this.group)
71        {
72          throw "no redoing while in a group!";
73        }
74        else if (this.redo_queue.length > 0)
75        {
76          var item = this.redo_queue.pop();
77
78          for (var fn in item.forward)
79          {
80            if (item.forward.hasOwnProperty(fn))
81            {
82              item.thing[fn] = item.forward[fn](item.thing[fn]);
83            }
84          }
85          console.log(item.thing);
86          this.undo_queue.push(item);
87          this.settings.onChange();
88        }
89      },
90
91    // internal functions
92    register_undo: function (n, f, b, t)
93      {
94        if (this.group)
```

```
95        {
96          this.undo_queue[this.undo_queue.length-1].name += ", " + n;
97          this.undo_queue[this.undo_queue.length-1].forward.push(f);
98          this.undo_queue[this.undo_queue.length-1].thing.push(t);
99          this.undo_queue[this.undo.length-1].backward = [b].concat(
100                 this.undo_queue[this.undo_queue.length-1].backward);
101       }
102       else
103       {
104         this.undo_queue.push({name: n, forward: [f], backward: [b], thing: [t]});
105       }
106     },
107
108   // internal state
109   undo_queue: [],
110   redo_queue: [],
111   group: false,
112
113 };
```

6.7 raphael-ext.js

```
1  /* Raphael extensions and utility functions
2   *
3   * Alex Burka, October 2011
4   */
5
6  /* Duplicate an object by transferring each property.
7   * TODO: should it recursively call deepcopy?
8   */
9  function deepcopy(from)
10 {
11   var to = {};
12
13   for (var k in from)
14   {
15     if (from.hasOwnProperty(k))
16     {
17       to[k] = from[k];
18     }
19   }
20
21   return to;
22 }
23
24 /*
25  * Select b if a is undefined.
26  *
27  * Just like a || b except it doesn't fail when a is 0/false.
28  */
29 function or(a, b)
30 {
31   return typeof a == "undefined" ? b : a;
32 }
33
34 /* Sets in Raphael are implemented in kind of a strange way. There seems to be no support
35  * in the actual SVG for the sets, they are just pseudo-arrays of the JS objects. Then any
36  * functions called on them are just mapped to each of the children with forEach. This is usually
37  * fine, but it means you can't store any data on the set itself. Also, it is hard or impossible
38  * to find out which set an element/set belongs to.
39  *
```

```
 * My extensions to Raphael.fn/st/el allow per-set data (by putting a _data element on
 * each set object) and also tracing through the set hierarchy, through el.parent()/st.parent().
 * This would break if something were to be added to two different sets.
 */
Raphael._sets = {}; // we need a record of all the sets created, indexed by ID
Raphael.fn._set = Raphael.fn.set;
Raphael.fn.set = function () { // store each new set in Raphael._sets, and give it a _data
  var s = this._set();
  s._data = {};

  var id = 0;
  for (var key in Raphael._sets) // get a unique ID
  {
    if (Raphael._sets.hasOwnProperty(key) && (key - 0) >= id)
    {
      id = (key - 0) + 1;
    }
  }

  Raphael._sets[id] = s;
  s.id = id;

  return s;
}
Raphael.fn.setById = function (id) { // grab a set out of Raphael._sets
  return Raphael._sets[id];
}
Raphael.st._push = Raphael.st.push; // mask Set.push, to set up the parent hierarchy
Raphael.st.push = function (/* ... */) {
  for (var i = 0; i < arguments.length; ++i)
  {
    if (arguments[i])
    {
      arguments[i]._parent = this.id;
      this._push(arguments[i]);
    }
  }
};
Raphael.st.parent = Raphael.el.parent = function ()
{
  return input.setById(this._parent);
}
Raphael.st.data = function (/* ... */) // Set.data, just like Element.attr
{
  switch (typeof arguments[0])
  {
    case "undefined": // no arguments: return all the data
      return this._data;

    case "string": // one property name
      switch (typeof arguments[1])
      {
        case "undefined": // one property name given: return the value
          return this._data[arguments[0]];
        default:
          this._data[arguments[0]] = arguments[1]; // name and value given: set it
          return this;
      }
      break;

    case "object": // dict given: set all the values
```

```
      for (var k in arguments[0])
      {
        if (arguments[0].hasOwnProperty(k))
        {
          this.data(k, arguments[0][k]);
        }
      }
      return this;
  }
}

// draw an arrow
//  parameters:
//    - x1/y1: coordinates of origin point
//    - x2/y2: coordinates of destination point (where the arrowhead is drawn)
//  returns: the arrow element
//  (all the examples online use atan2 to draw a rotated triangle
//   but there is an easy Raphael attribute to add an arrowhead)
//  note: the returned element is a path, but its type is set to "arrow"
//    this has been observed to cause problems occasionally
//      (like when using Raphael to modify the path string later)
//      if it does, just change the type back to "path"
Raphael.fn.arrow = function (x1, y1, x2, y2) {
  var arrow = this.path("m" + x1 + "," + y1
            + "l" + (x2-x1) + "," + (y2-y1));

  arrow.attr("arrow-end", "block-wide-long");
  this.type = "arrow";
  return arrow;
}
```

Diagramr

A Web-based Automatic Shear/Bending Moment Diagrammer

Alex Burka

ENGR 059 Final Project
Final Presentation
December 5, 2011

Outline

- Acknowledgements
- Project goals and requirements
- Theoretical foundations
- Implementation
- DEMO
- Future work

Acknowledgements

- Professor Siddiqui
- Ames Bielenberg
- Raphaël developer (Dmitry Baranovskiy)
- Firefox developers

Project Goals

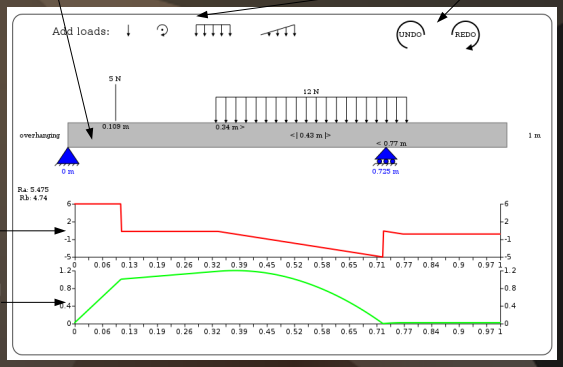
- Pedagogical tool for introductory mechanical engineering
- Create shear and bending moment plots from beam loading diagrams
  - Singularity functions
- Solve real textbook problems

Requirements

- Intuitive interface
  - Must be easier to use the program than to work the problem by hand
- Correct answers
  - The program must detect nonsensical input rather than give incorrect output
- Speed
- Portability

Statically determinate beam

Controls



Shear force

Bending moment

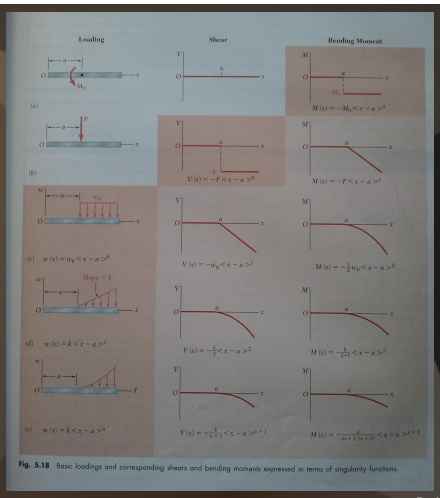


Fig. 5.18 Basic loadings and corresponding shears and bending moments expressed in terms of singularity functions.

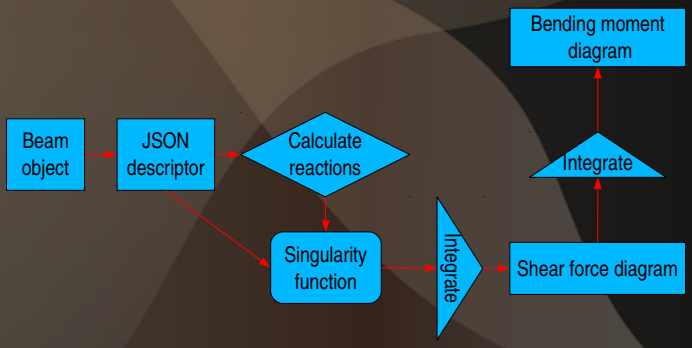
Singularity Functions

- Express state of entire beam in one equation
- Singularity function includes a term for each beam support or load
- Each term is multiplied by a step function to specify its point of action: $\langle x-a\rangle^n=(x-a)^n u(x-a)$
- Easy integration rules: $\int \langle x-a\rangle^n dx = \begin{cases} \langle x-a\rangle^{n+1}, & n \le 0 \\ \frac{\langle x-a\rangle^{n+1}}{n+1}, & n \ge 0 \end{cases}$
- Distributed loads always extend to infinity, so finite loads must be counterbalanced

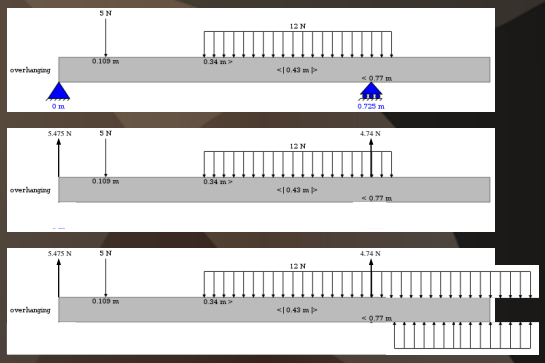
Implementation

- Web application
  - Interface and processing written in JavaScript
    - All code runs on the client side
      - URI used for local storage
    - Beam graphics by Raphaël.js (SVG)
    - Graphs by g.Raphaël.js (also SVG)

Data flow



Example



$w(x) = -5.475\langle x\rangle^{-1} + 5\langle x-.109\rangle^{-1} + 12\langle x-.34\rangle^{0} - 12\langle x-.77\rangle^{0}$

Future work

- Cross-browser compatibility
- Generic loads
- Statically indeterminate problems
- User's guide

Lessons Learned

- Solid understanding of beam loading
- Computers are helpful, but so is common sense
- This webpage may need to be password-protected during E6 semesters

References

- Beer et al. *Mechanics of Materials.* 6 ed. 2012
- http://www.raphaeljs.com